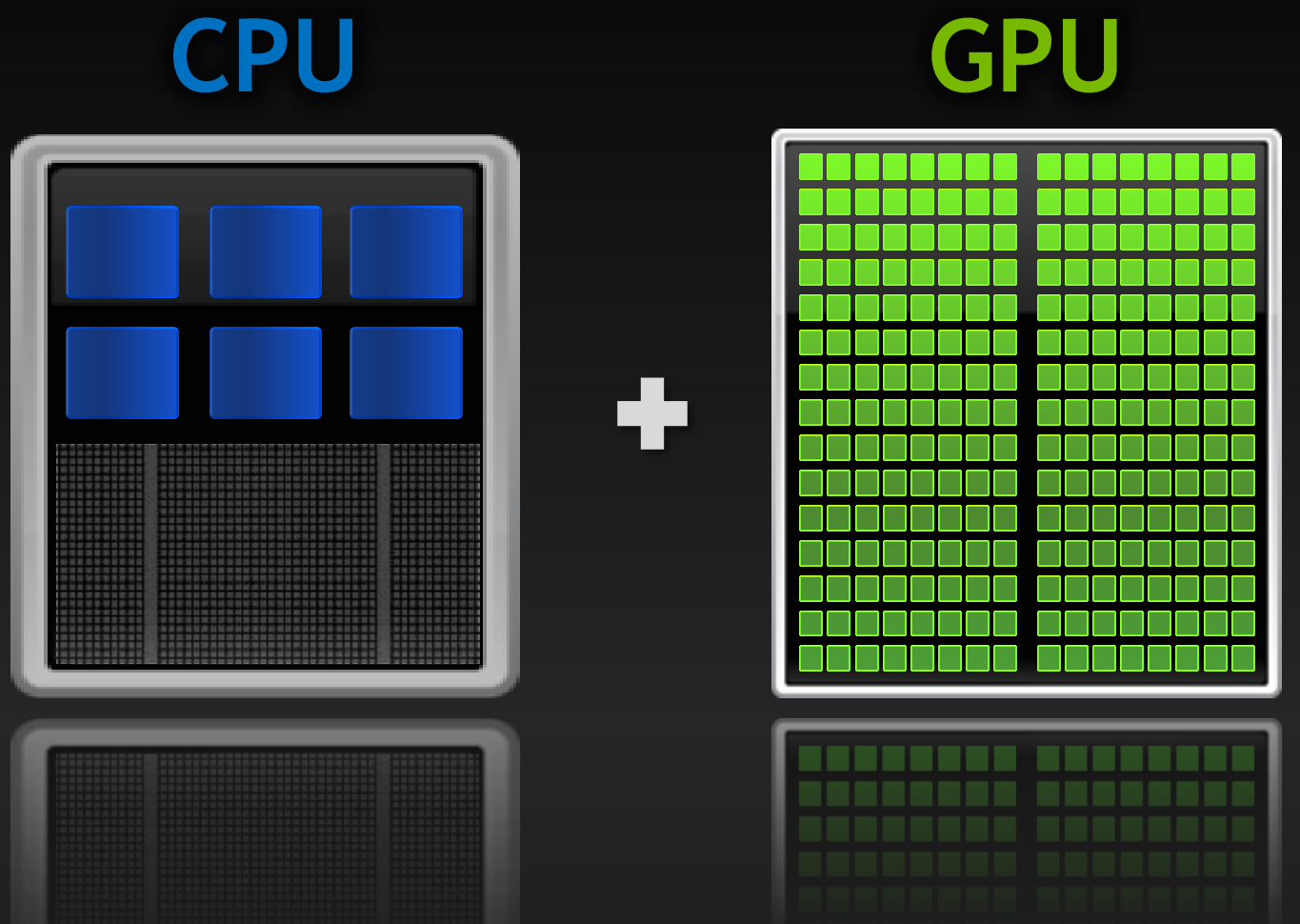
Использование GPU в Графовых Задачах

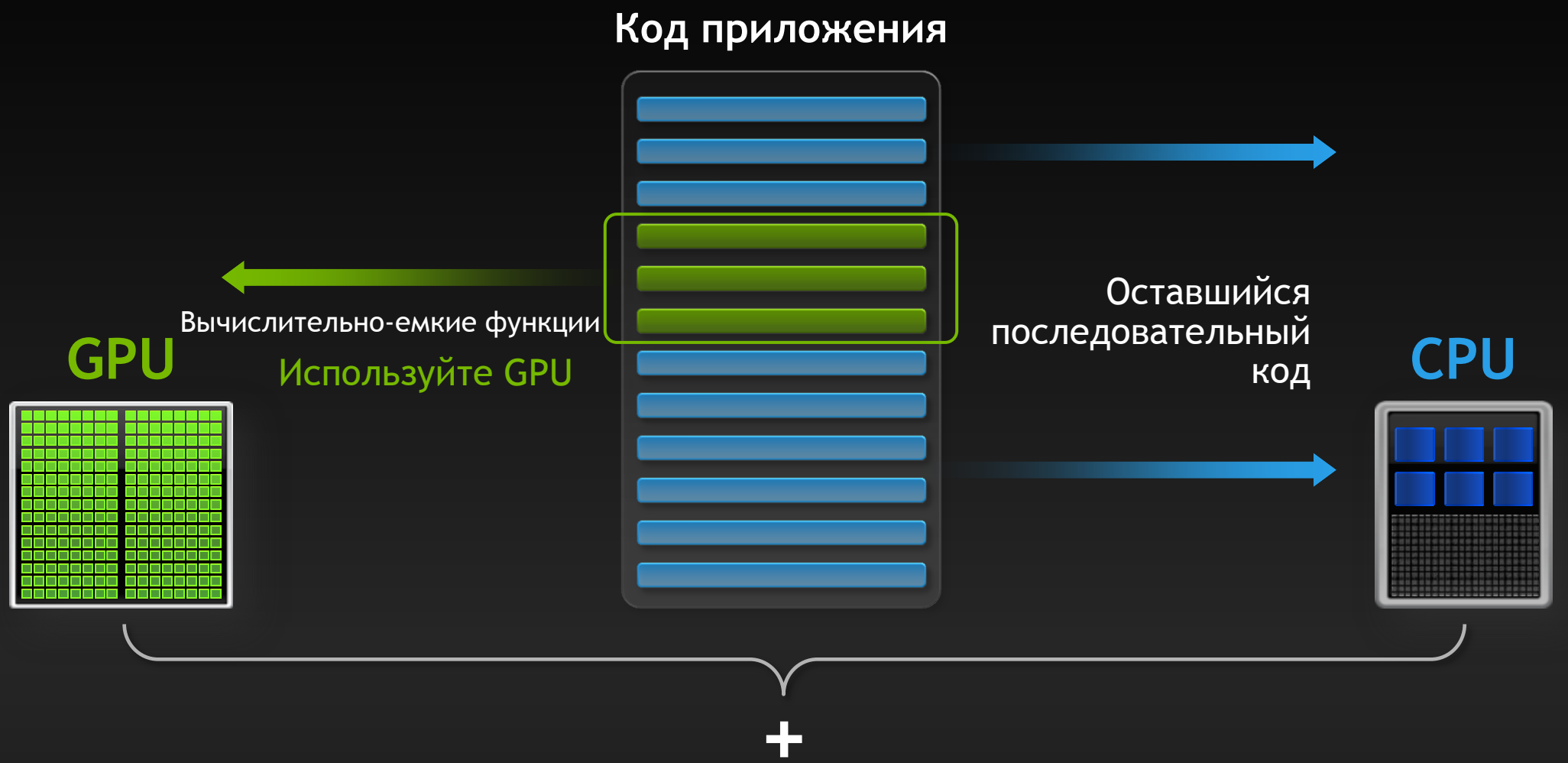
GraphHPC-2014

Максим Милаков, NVIDIA

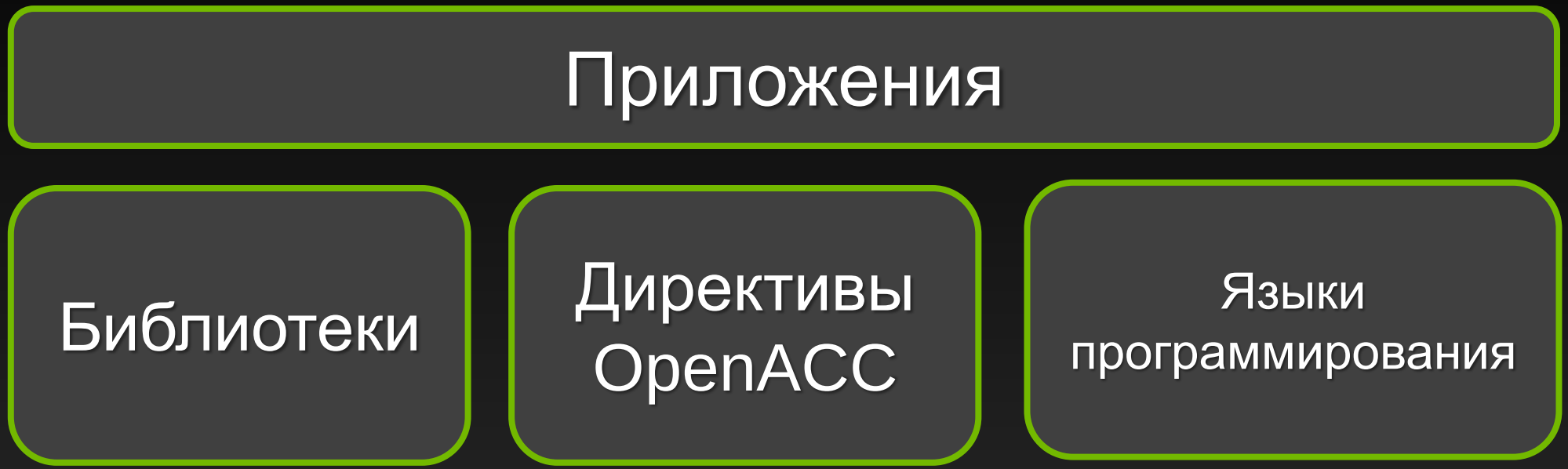
Добавьте GPU: ускорьте ваши приложения



Небольшие изменения, существенное ускорение

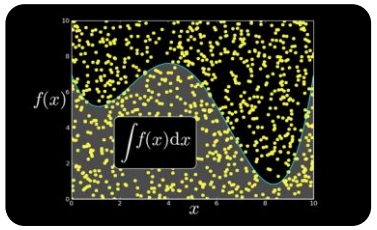


3 способа ускорения приложений

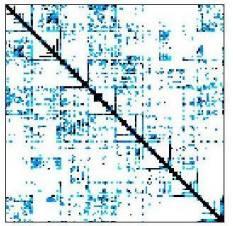


Некоторые библиотеки для работы с GPU

NVIDIA CUBLAS



NVIDIA CURAND



NVIDIA CUSPARSE

NVIDIA NPP

GPU VSIPL

Vector Signal Image Processing

CULA tools

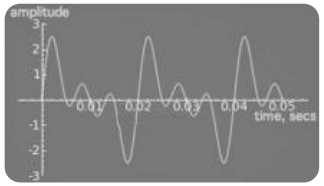
GPU Accelerated Linear Algebra

MAGMA

ICL UT

Matrix Algebra on GPU and Multicore

open source initiative



NVIDIA CUFFT

ROGUE WAVE SOFTWARE

IMSL Library

ArrayFire Matrix Computations

CUSP

Sparse Linear Algebra

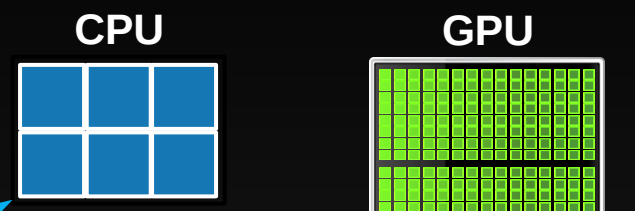
open source initiative

Thrust

C++ STL Features for CUDA

open source initiative

Директивы OpenACC




```
Program myscience
   ... последовательный код ...
!$acc kernels
   do k = 1,n1
      do i = 1,n2
         ... параллельный код ...
      enddo
    enddo
!$acc end kernels
  ...
End Program myscience
```

OpenACC “подсказки” компилятору

Ваш исходный Fortran или C код

Простые “подсказки” компилятору

Компилятор параллелизует код

Код выполняется на ускорителе

Языки программирования GPU

Numerical analytics	MATLAB, Mathematica, LabVIEW
Fortran	OpenACC, CUDA Fortran
C	OpenACC, CUDA C
C++	Thrust, CUDA C++
Python	PyCUDA, Copperhead
C#	CUDAfy.NET, Hybridizer

CUDA C

Обычный код C

```
void saxpy_serial(int n,
                  float a,
                  float *x,
                  float *y)
{

  for (int i = 0; i < n; ++i)
    y[i] = a*x[i] + y[i];
}

// Perform SAXPY on 1M elements
saxpy_serial(4096*256, 2.0, x, y);
```

Параллельный код C

```
__global__
void saxpy_parallel(int n,
                    float a,
                    float *x,
                    float *y)
{
  int i = blockIdx.x*blockDim.x +
          threadIdx.x;
  if (i < n) y[i] = a*x[i] + y[i];
}

// Perform SAXPY on 1M elements
saxpy_parallel<<<4096,256>>>(n,2.0,x,y);
```

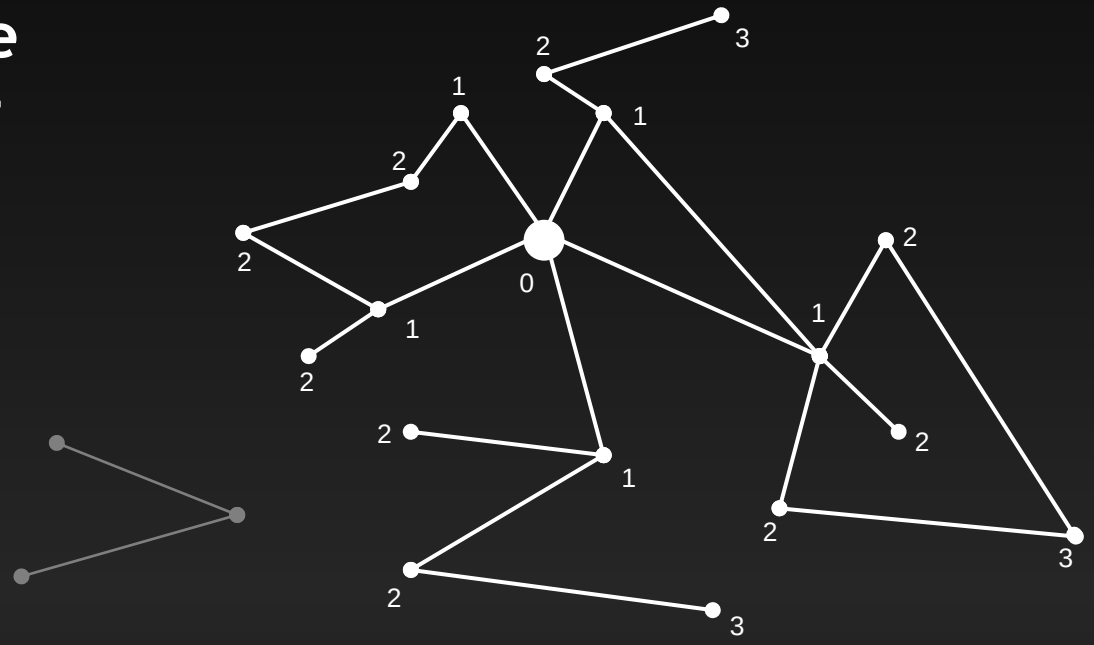
http://developer.nvidia.com/cuda-toolkit

Графовые задачи на GPU

- Поиск в ширину
- Дейкстра, A*, R*, Beam
- Наибольшее независимое множество
- Минимальное остовное дерево

BFS - Поиск в ширину

- Выбрать вершину-источник
- Присвоить каждой вершине длину кратчайшего пути от источника



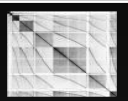
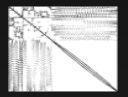
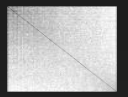
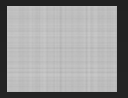
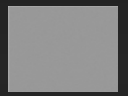
Поиск в ширину: Приложения

- Достижимость (например, при сборке мусора)
- Алгоритм распространения доверия (в графических вероятностных моделях)
- Определение структуры сообщества
- Нахождение пути
- Graph 500 бенчмарк

Масштабируемый обход графа

- Duane Merrill, Michael Garland, Andrew Grimshaw, Scalable GPU graph traversal
  - Статья - http://dl.acm.org/citation.cfm?id=2145832
  - Презентация - http://on-demand.gputechconf.com/gtc/2012/presentations/S0600-Scalable-GPU-Graph-Traversal.pdf
  - Видео - http://on-demand.gputechconf.com/gtc/2012/video/S0600-Scalable-GPU-Graph-Traversal.mp4
  - Исходный код - https://code.google.com/p/back40computing/
- Скорость работы - более 3.3 миллиардов and 8.3 миллиардов ребер в секунду используя соответственно один GPU и 4x GPU конфигурации (Tesla C2050)

Сравнение с CPU

Граф	Матрица разреженности	Средняя глубина посика	Nehalem		Tesla C2050	
			Миллиарды ребер в секунду	Parallel speedup	Миллиарды ребер в секунду	Parallel speedup†††
europe.osm		19314			0.3	11 x
grid5pt.5000		7500			0.6	7.3 x
hugebubbles		6151			0.4	15 x
grid7pt.300		679	0.12 (4-core†)	2.4 x	**1.1**	28 x
nlpkkt160		142	0.47 (4-core†)	3.0 x	2.5	10 x
audikw1		62			3.0	4.6 x
cage15		37	0.23 (4-core†)	2.6 x	2.2	18 x
kkt_power		37	0.11 (4-core†)	3.0 x	1.1	23 x
coPapersCiteseer		26			3.0	5.9 x
wikipedia-2007		20	0.19 (4-core†)	3.2 x	1.6	25 x
kron_g500-logn20		6			3.1	13 x
random.2Mv.128Me		6	0.50 (8-core††)	7.0 x	3.0	29 x
rmat.2Mv.128Me		6	0.70 (8-core††)	6.0 x	3.3	22 x

† 2.5 GHz Core i7 4-core (Leiserson *et al.*) †† 2.7 GHz EX Xeon X5570 8-core (Agarwal *et al*) ††† vs 3.4GHz Core i7 2600K (Sandybridge)

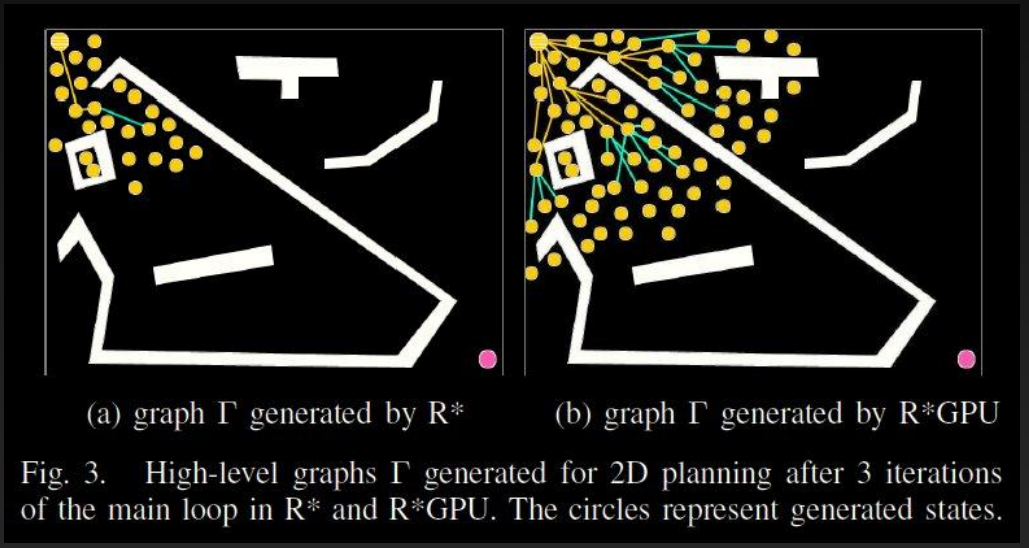
Алгоритм Дейкстра, A*

- Алгоритмы нахождения кратчайшего расстояния от одной вершины графа до всех остальных
  - A* - с эвристикой
- Доступно большое количество реализаций на GPU (различной степени успешности) и сопутствующих статей

R*

- R* - рандомизированная версия A*, эффективно работающая в многомерных задачах планирования

- Хорошо параллелизуется
- Joseph T. Kider Jr., Mark Henderson, Maxim Likhachev, and Alla Safonova, High-dimensional Planning on the GPU - http://hgpu.org/?p=3203



(a) graph Γ generated by R* (b) graph Γ generated by R*GPU

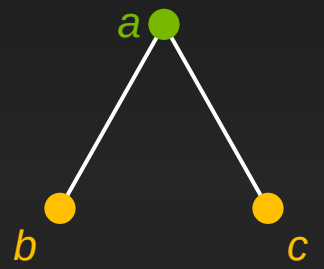
Fig. 3. High-level graphs Γ generated for 2D planning after 3 iterations of the main loop in R* and R*GPU. The circles represent generated states.

Beam search

- Обобщение A*/BFS – ширина фронта ограничена
- Используется в распознавании речи
  - И вообще везде, где необходимо получить скорость, пусть и в ущерб качеству
- Реализации:
  - Patrick Kano, A CUDA accelerated Beam Propagation Method [BPM] Solver using the Parallel Computing Toolbox, MATLAB, http://www.mathworks.com/matlabcentral/fileexchange/29114-a-cuda-accelerated-beam-propagation-method-bpm-solver-using-the-parallel-computing-toolbox
  - Jungsuk Kim, Kisun You and Wonyong Sung, H- and C-Level WFST-based Large Vocabulary Continuous Speech Recognition On Graphics processing Units
  - Jungsuk Kim, Jike Chong, Ian Lane, Efficient On-The-Fly Hypothesis Rescoring in a Hybrid GPU/CPU-based Large Vocabulary Continuous Speech Recognition Engine, http://www.cs.cmu.edu/~ianlane/publications/2012_Kim_Interspeech.pdf

Наибольшее независимое множество

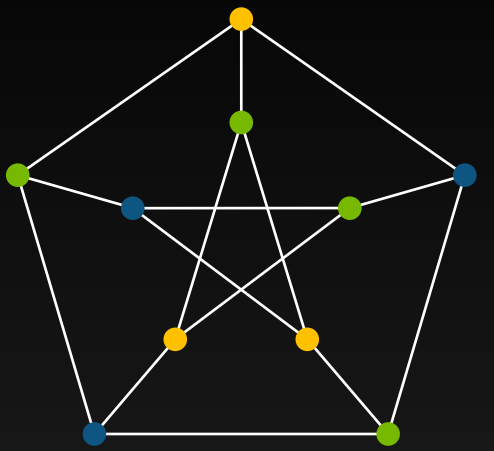
- Подмножество вершин $U \in V$ неориентированного графа $G = (V, E)$
  - U - *независимое*, если никакие две вершины U не соеденины ребром из E
  - U - *наибольшее*, если никакая вершина из V не может быть добавлена к U без нарушения условия независимости

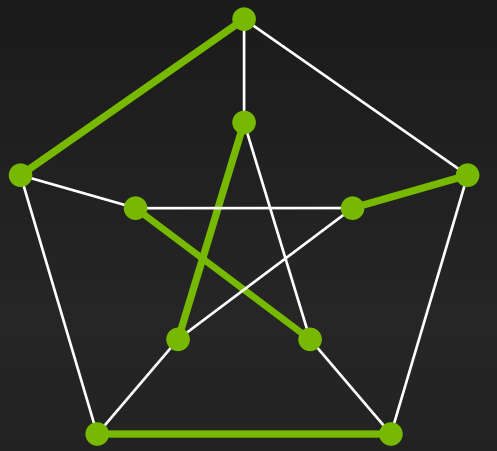


Как $\{a\}$, так и $\{b, c\}$ являются наибольшими независимыми множествами

MIS: Приложения

- Раскраска графов (вершинная)
  - Решение СЛУ с разреженной матрицей (включая алгебраический многосеточный метод)
  - Моделирование поведения твердых тел
- Паросочетание (раскраска ребер)
  - Попарное связывание
    - Планирование выполнения задач на серверах
    - Рассадка людей за столы, сайт знакомств и т.д.
  - Найти MIS в графе G’:
    - Вершина в G’ для каждого ребра в G
    - Вершины соеденины в G’ соответствующие ребра в G смежны



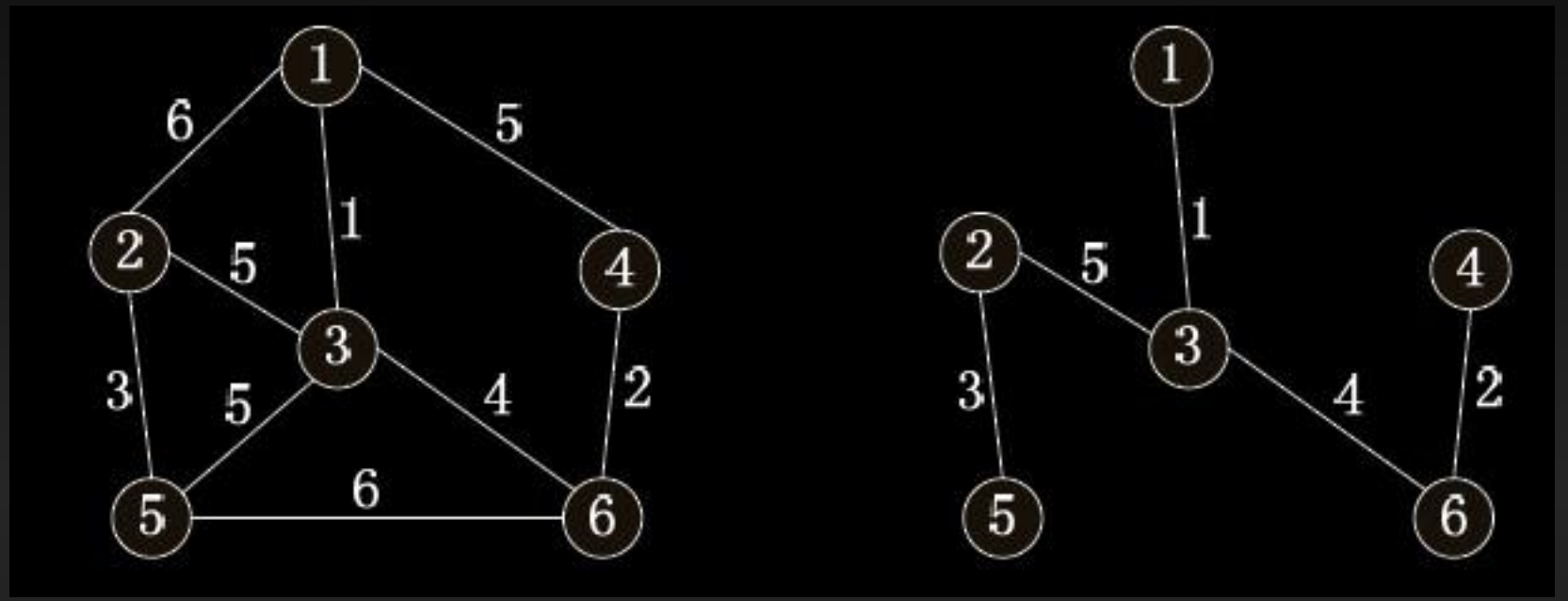


MIS: Статьи

- Patrice Castonguay, Jonathan Cohen, Efficient Graph Matching and Coloring on the GPU
  - Презентация - http://on-demand.gputechconf.com/gtc/2012/presentations/S0332-Efficient-Graph-Matching-and-Coloring-on-GPUs.pdf
  - Видео - http://on-demand.gputechconf.com/gtc/2012/video/S0332-Efficient-Graph-Matching-and-Coloring-on-GPUs.mp4
- Cristina Nader Vasconcelos and Bodo Rosenhahn, Bipartite Graph Matching Computation on GPU - http://www.tnt.uni-hannover.de/papers/data/780/780_1.pdf, 7x

Минимальное остовное дерево

- Остовное дерево неориентированного графа, имеющее минимально возможный вес



MST: реализации

- Параллельный алгоритм Прима - Wei Wang, Yongzhong Huang, and Shaozhong Guo, Design and Implementation of GPU-Based Prim's Algorithm - http://www.mecs-press.org/ijmecs/ijmecs-v3-n4/IJMECS-V3-N4-8.pdf
- Параллельный алгоритм Борувки - Vibhav Vineet, Pawan Harish, Suryakant Patidar, P. J. Narayanan, Fast Minimum Spanning Tree for Large Graphs on the GPU - http://cvit.iiit.ac.in/papers/Vibhav09Fast.pdf, 30x-50x
- Применение: например, прокладка кабелей телекоммуникационной компанией


Спасибо!
developer.nvidia.com
mmilakov@nvidia.com
NVIDIA